JAPAN | JUNE 21, 2024

aws SUMMIT

フィーチャーフラグの現在（いま）と未来

Hiroto Sakuraya

Senior Partner Solutions Architect, SaaS
Amazon Web Services Japan G.K.

自己紹介

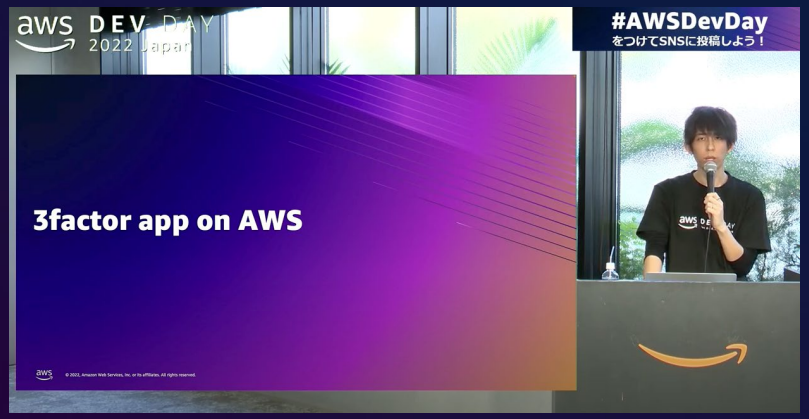
さくらや　ひろと

櫻谷 広人

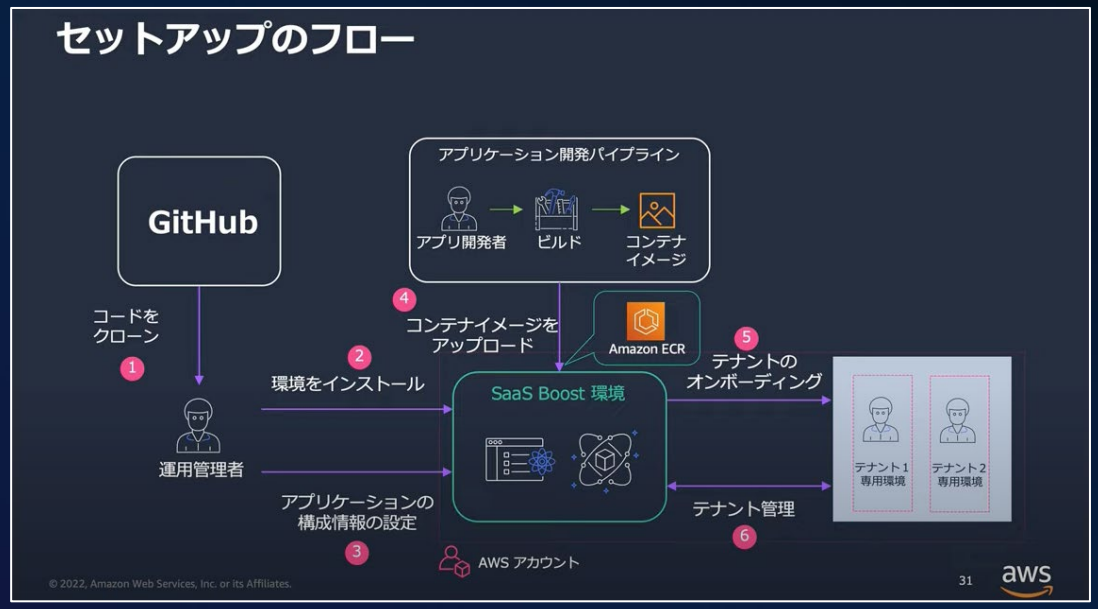
Partner Solutions Architect, SaaS
Amazon Web Services Japan G.K.

得意分野

SaaS, Serverless
マイクロサービスとかマルチテナントとか



[3factor app って結局のところ何なの？ | AWS Dev Day 2022 Japan #AWSDevDay]



[AWS BlackBelt Online Seminar] AWS SaaS Boost で始める SaaS 開発入門

このセッションの目的

言葉としては聞いたこともあるし、どんなものかはなんとなーくイメージができるけど、実際は奥が深いフィーチャーフラグ。

- 2024年現在、トレンドとしてはどうなのか？
- 簡単に導入できるものなのか？
- そもそもなんで必要なのか？
- これから何をすればいいのか？

今から真剣にフィーチャーフラグと向き合ってみましょう

今日お話しすること

- フィーチャーフラグって何だろう
- フィーチャーフラグの定義とその効果
- フィーチャーフラグを実装するには
- まとめ

フィーチャーフラグってなんだろう？

機能の有効化/無効化するやつだよね...？

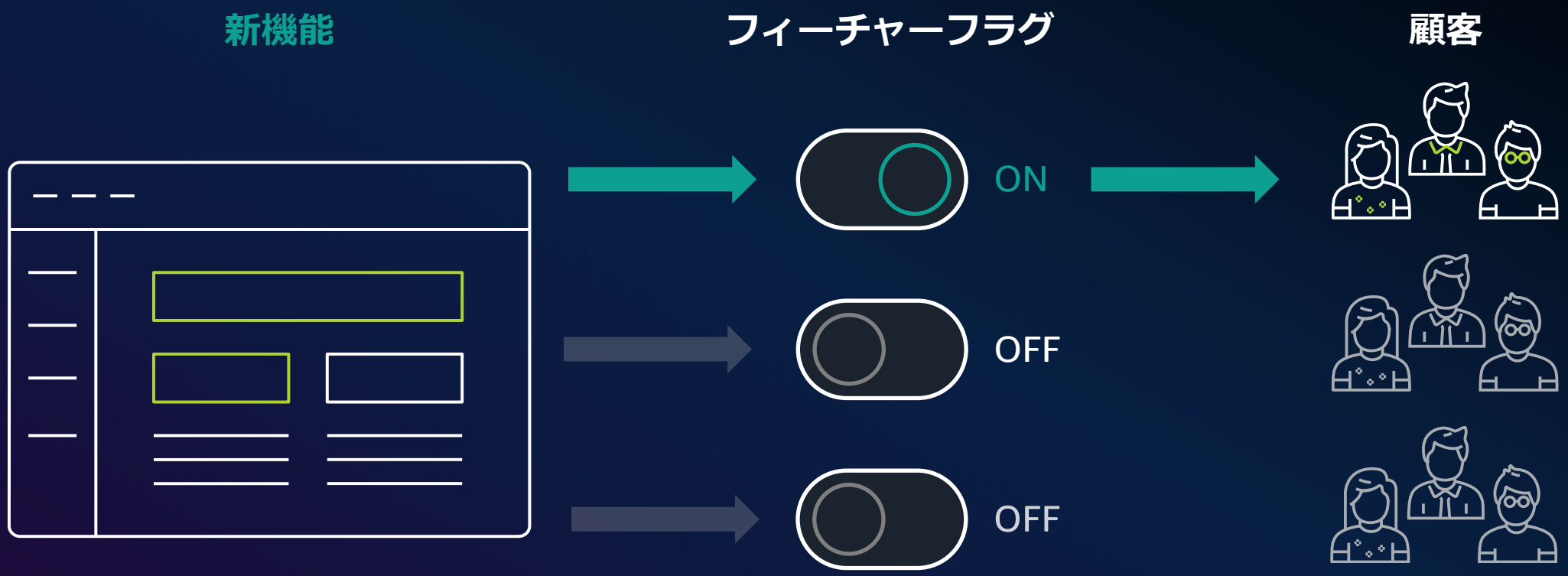
A/B テストとかで使う特殊なやつだった気が...？

if / else のコードのこと？

なんとなくの
イメージ→

トグルスイッチ: レバーを一方向に倒すことで電気回路を切り変えるスイッチ

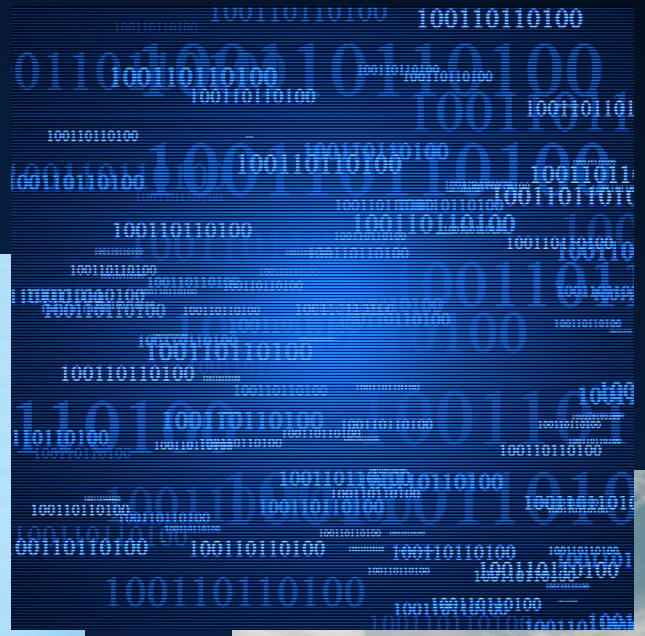
SaaS でよく使われてるやつだ



呼び方は色々あるが...

※ このセッションではフィーチャーフラグと呼んでいきます

Feature Flag / Feature Toggle / Feature Bits / Feature Flippers



呼び方は色々あるが...

Feature Flag / Feature Toggle / Feature Bits / Feature Flippers

俺たちはまだフィーチャーフラグの
ことをよく知らない

フィーチャーフラグのバイブル

https://martinfowler.com/articles/feature-toggles.html

> Feature Toggles (often also refered to as Feature Flags) are a powerful technique, allowing teams to modify system behavior without changing code.
>
> *Pete Hodgson*

フィーチャーフラグとは、コードを変更することなく
システムの振る舞いを変えることができる強力なテクニックである

Feature Toggles (often also refered to as Feature Flags) are a powerful technique, allowing teams to modify system behavior without changing code.

Pete Hodgson

フィーチャーフラグとは、コードを変更することなく
システムの振る舞いを変えることができる強力なテクニックである

システムの振る舞いって？

例）ボタンを押したら、ポップアップが表示される

このユーザーからのリクエストは、拒否する

5 回ログインに失敗したら、アカウントをロックする

0～4時にアクセスが来たらメンテナンスページを表示する

検証環境のデータベースに接続する

システムの振る舞いを変えるには？

例）ボタンを押したら、ポップアップが表示される

HTML ファイル内のJavaScript

このユーザーからのリクエストは、拒否する

Users テーブルのカラム

5 回ログインに失敗したら、アカウントをロックする

0～4時にアクセスが来たらメンテナンスページを表示する

検証環境のデータベースに接続する

IDaaS 内の設定

env ファイル

nginx の config

> Feature Toggles (often also refered to as Feature Flags) are a powerful technique, allowing teams to modify system behavior without changing code.
>
> *Pete Hodgson*

フィーチャーフラグとは、**コードを変更することなく**
システムの振る舞いを変えることができる強力なテクニックである

Feature Toggles (often also refered to as
Fe
al
be

Pet

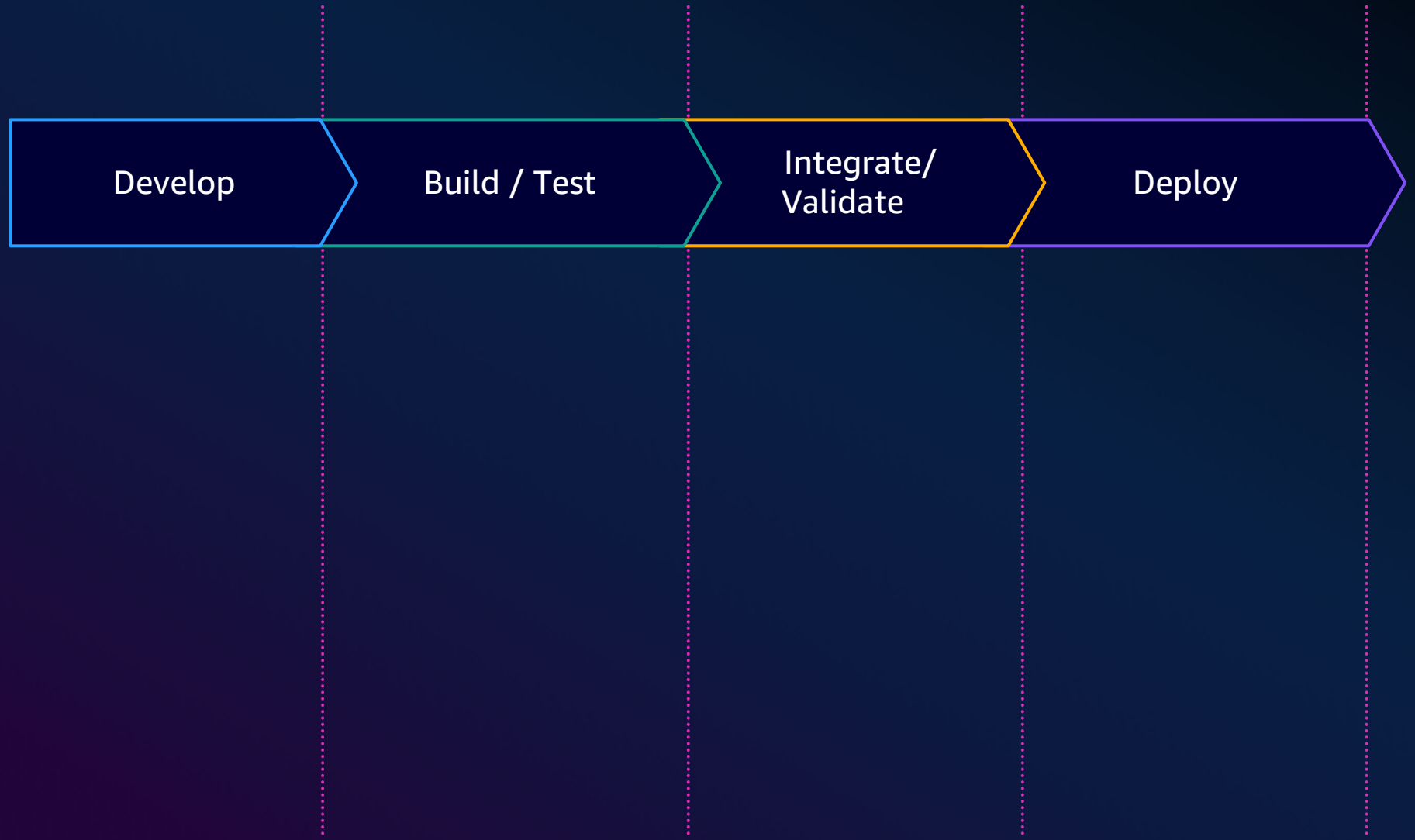
俺たちはもっと手軽に
リリースを行いたい

システムの振る舞いを変えることができる強力なテクニックである

Deployment is not Release

フィーチャーフラグがない世界

- Deploy = Release
- 全ユーザーに一斉公開
- ロールバックも再デプロイ

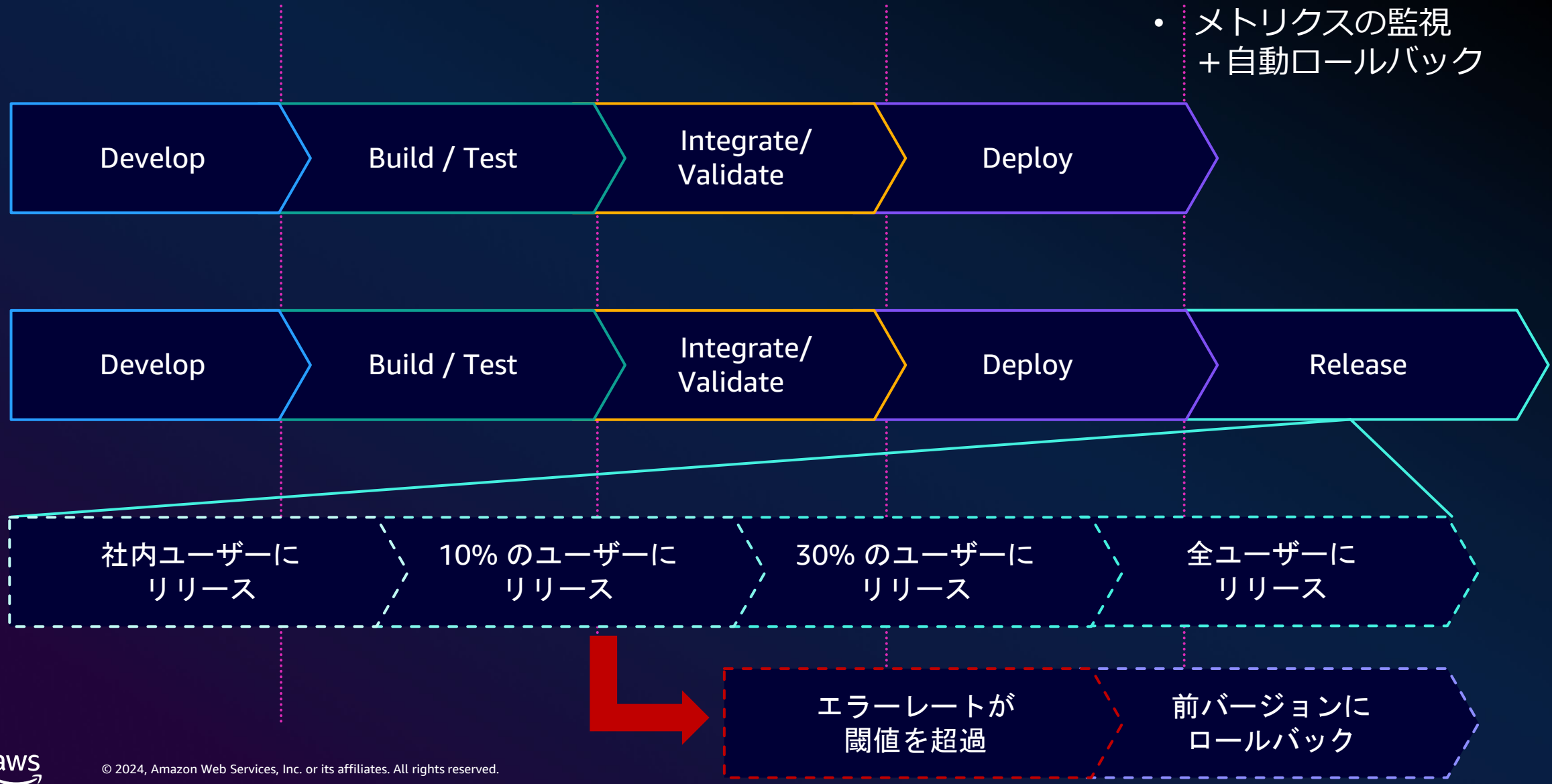


フィーチャーフラグ導入初期

- 開発中のコードを本番環境にデプロイ
- フラグによって隠されている状態
- 準備が整ったらリリース
- 作業はフラグを変えるだけ

フィーチャーフラグを使いこなす

- 社内ユーザーのみでのテスト
- ベータユーザーでのテスト
- 段階的デプロイ
- メトリクスの監視
  ＋自動ロールバック



フィーチャーフラグの本質

- コードを変更することなく、システムの振る舞いを変えるための手段
- より高速に、より安全に、動的に振る舞いを変えることができる
- これにより、気軽に多くの**実験**ができるようになる
- 結果として、**ビジネスイノベーションの加速**につながる

フィーチャーフラグを導入すると

- 特定のコンテキストにおける機能の有効/無効化 (≒ Allow/Deny)
  (ユーザー、テナント、リクエスト、環境、etc.)
- A/B テスト、ベータテスト
- カナリアデプロイ、迅速なロールバック
- トランクベース開発

etc.

フィーチャーフラグの種類

- Release Flag: 開発中の新機能、テストされてない不完全なコードを本番環境にデプロイしつつ、ユーザーには隠すためのフラグ
- Ops Flag: 運用タスクで利用するためのフラグ。例えば、負荷が高くなった時に、重要でない機能をオフにしてサービスを安定させるなど
- Experimental Flag: ユーザーをランダムにグループ分けして別の振る舞いをさせるなど、A/B テストなどでよく使われるフラグ
- Permissions Flag: ベータユーザーへのアーリーアクセスの提供、マルチテナント SaaS における契約ティアに応じた機能の開放などに使われるフラグ

“もしも”のための非常停止ボタンとして

フィーチャーフラグを実装するには

フィーチャーフラグ: レベル 1

```
const featureIsEnabled = true
function superFeature() {
  if (featureIsEnabled) {
    // allow use this feature
  } else {
    // deny use this feature
  }
}
```

- フラグの更新にはコードの変更が必要
- フラグの更新のたびにデプロイが発生
- ロールバックも同様
- 段階的ロールアウトが不可能

etc.

そもそもこれは**フィーチャーフラグ**と呼べるのだろうか…？

フィーチャーフラグの実装でまず検討すべきこと

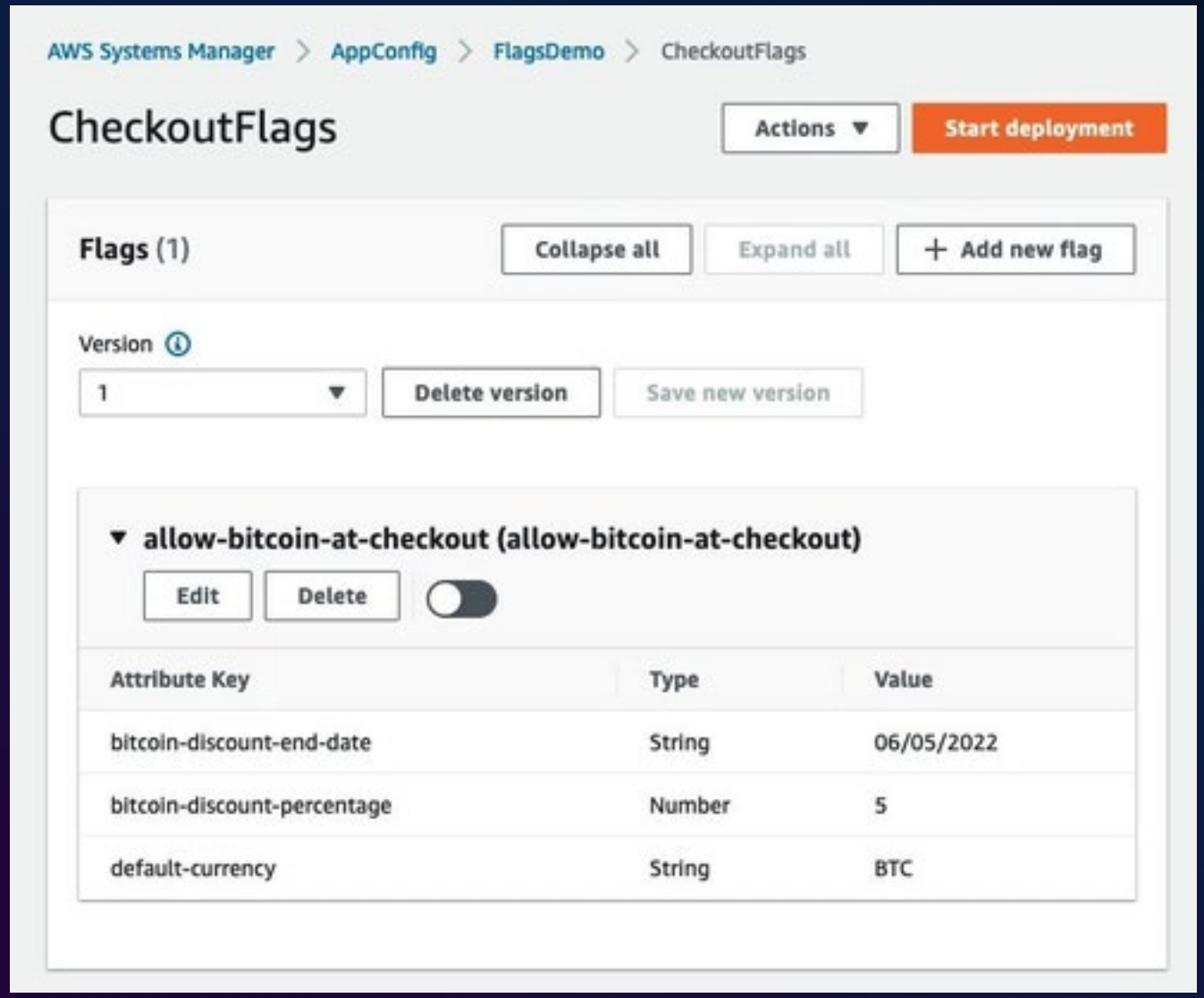
重要なのは、どこでどのようにフラグを**管理**するか

(例: アプリケーションコード、パラメータ、設定ファイル、データベース、インメモリ、外部サービス、etc.)

どう管理するかによって、パフォーマンス、スケーラビリティ、可用性、運用効率、更新のしやすさなどに違いが出てくる

フラグは動的なのか静的なのか、更新頻度はどれくらいか、
生存期間はどれくらいか、などのユースケースも整理する

フィーチャーフラグ: レベル 99



- フラグの管理は外部サービスにお任せ
- クライアント側から呼び出すだけ
- Web UI からフラグ更新
- 便利なマネージド機能
  - バージョニング
  - バリデーション
  - カナリアデプロイ
  - CI/CD パイプラインとの統合

etc.

あくまで一例

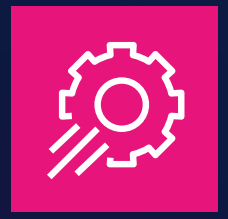
車輪は再発明するな

Feature Flag as a Service

- まずはサードパーティの製品が使えないか検討する
  - 必要な機能を備えているか (バージョニング、バリデーション、段階ロールアウト、リアルタイム更新、スケジューリング、Web コンソール、etc)
  - 非機能要件を満たせるか (パフォーマンス、可用性、コスト、セキュリティ、etc)
  - SDK は提供されているか

ex. CloudBees, Optimizely, LaunchDarkly, Split, Unleash, Statsig, ConfigCat

AWS のサービスも！→



AWS AppConfig

Amazon CloudWatch
Evidently

フィーチャーフラグの未来

- ベンダーに依存しない、共通のインターフェースを持ったフィーチャーフラグの**オープン標準**を作ることを目指した OpenFeature というプロジェクトがある
- コミュニティ主導
- CNCF incubating project (2023年末に sandbox から昇格)
- 様々な言語向けに SDK を提供
  (ex. Go, Java, .NET, Node.js, PHP, Python, Android, iOS, React)
- フィーチャーフラグのバックエンドシステムとして動く flagd など周辺ツールも
- 仕様はまだ fix していないが、すでにサポートを始めるベンダーも

OpenFeature

https://openfeature.dev/

フィーチャーフラグにリスクはないのか？

もちろんある

- 杜撰な運用で意図せず未発表の機能を公開してしまったり
- 無効な値を入れてアプリをバグらせてしまったり
- 古いフラグが残っていて意図せぬ動作を引き起こしてしまったり
- フラグ同士が密結合になったり

同時に運用しているフラグの数が多くなれば全体像は複雑になり、
全ての組み合わせパターンを網羅したテストも難しくなる。

ただしこれは**コードと同じ**で、事前のレビューやテスト、リリース管理が鍵

まとめ

- フィーチャーフラグは、開発者の生産性を上げ、ビジネスイノベーションを加速させるための方法
- 機能の有効化/無効化だけでなく、運用の効率化やユーザーエクスペリエンスの向上にも使える
- 専門の Feature Flag as a Service も増えてきている
- OpenFeature によるベンダー非依存のオープン標準が作られる動きも

Thank you!